Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Janne Uggeldahl

Puolijohdereleiden testilaitteiston toteutus LabVIEW-ohjelmointiympäristössä

Insinöörityö 17.11.2009
Ohjaaja: tuotekehityspäällikkö Reima Konttinen
Ohjaava opettaja: lehtori Anssi Ikonen

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä Otsikko Sivumäärä Aika	Janne Uggeldahl Puolijohdereleiden testilaitteiston toteutus LabVIEW -ohjelmointiympäristössä 47 sivua 17.11.2009
Koulutusohjelma	tietotekniikka
Tutkinto	insinööri (AMK)
Ohjaaja Ohjaava opettaja	tuotekehityspäällikkö Reima Konttinen lehtori Anssi Ikonen

Insinöörityön tehtävänä oli suunnitella ja toteuttaa puolijohdereleiden testilaitteisto LabVIEW-ohjelmointiympäristössä. Työ tehtiin Delcon Oy:lle vanhasta testilaitteistosta saatujen kokemuksien pohjalta. Helppokäyttöisyys ja luotettavuus olivat sovelluksen tärkeimmät vaatimukset.

Uuden laitteiston oli määrä poistaa vanhaa kokoonpanoa häirinneet mittausvirheet sekä nopeuttaa testausprosessia. Mittaustapahtuman raja-arvot tuli voida lukea taulukosta ja mitatut arvot tallentaa vastaavasti taulukkomuotoon. Raja-arvotaulukoiden lisääminen testilaitteistoon oli määrä tehdä helpoksi, jotta uusien tuotteiden testaaminen onnistuisi vaivattomasti. Testilaitteiston tuli olla tietotekniikkaa osaamattomalle peruskäyttäjälle mahdollisimman helppokäyttöinen sekä toimintavarma.

Testilaitteistoon valittiin USB-liitäntäiset mittauskortit, jotta laitteiston siirrettävyys olisi mahdollisimman hyvä tietokoneesta toiseen. Sovelluksessa käytettiin Master & Slave -rakennetta, jolla saatiin erotettua käyttöliittymä- ja testausosiot luontevasti toisistaan. Edellisestä testilaitteistosta poiketen jännitelähteet mitoitettiin siten, että laitteistolla saatiin testattua kaikki Delcon Oy:n valmistamat puolijohdereleet.

Lopullinen testilaitteiston versio jäi raja-arvojen hienosäätöä vaille valmiiksi. Ylikuulumisesta ja mittauskytkennästä johtuvien rajoitteiden vuoksi testilaitteistosta ei saatu viritettyä tarkkuusmittalaitetta. Toiminnallisuus rajoittui viallisten tuotteiden havaitsemiseen, sillä yksilökohtaisia eroja ei saatu luotettavasti mitattua. Peruskäyttäjälle oleellisen virheellisten tuotteiden havaitsemisen laitteisto hoitaa kuitenkin sekä nopeasti että luotettavasti. Testattava tuote sekä erätunnus olisi tulevaisuudessa mahdollista lukea järjestelmään työmääräimen viivakoodista erillisen viivakoodinlukijan avulla. Näin päästäisiin eroon inhimillisestä virheestä, joka aiheutuu käyttäjän virhelyönneistä näppäimistöllä.

Hakusanat	LabVIEW, puolijohderele, testilaitteisto, DAQ

Helsinki Metropolia University of Applied Sciences **Abstract**

Author Title Number of Pages Date	Janne Uggeldahl Test bench development with LabVIEW for solid state interface relays 47 pages 17 November 2009
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Instructor Supervisor	Reima Konttinen, Product Development Manager Anssi Ikonen, Principal Lecturer

The purpose of this thesis was to design and implement a test bench for solid state interface relays. The project was done for Delcon Oy using the feedback provided by the users of the old test bench. Usability and reliability were the key requirements for the application.

The new test bench was designed to get rid of measurement errors and to speed up the testing procedure. The limiting values used in testing had to be read from a tab delimited text file or a spreadsheet. Also the measured data had to be saved in spreadsheet form for later use. The adding of new spreadsheets containing the limiting values were to be made easy so new products could be added effortlessly to the test bench by the product development department in the future. The application had to be usable and reliable when operated by basic level users.

USB-based data acquisition hardware was chosen to maximize portability from one workstation to another. A design hierarchy called the Master & Slave structure was implemented to separate the interface and testing segments of the application. The power sources of the system were carefully chosen so that all of the Delcon-relays could be tested efficiently.

Final version of the test bench application included full functionality, only the adjustment of limit values was left to the product development department. Due to unexpected crosstalk and limitations of the measurement circuit, the test bench could not be used as a precision measurement device. The key requirement of detecting faulty products was implemented successfully. In the future the test bench could be implemented with a barcode reader to minimize the chance of human error occurring at the input of the product type and ID.

Keywords	LabVIEW, solid state relay, test bench, DAQ

Lyhenne- ja käsiteluettelo

A/D-muunnin	laite, joka muuntaa jatkuvan analogiasignaalin digitaalisiksi lukuarvoiksi näytteistämällä
DAQ	tiedonkeruukortti (eng. Data Acquisition device)
Diagrammi	LabVIEW'n graafisesta koodista koostuva osio (eng. Block Diagram)
Indikaattori	objekti, joka tulostaa tietoa etulevylle, esim. graafi tai merkkivalo
ISA-väylä	Industrial Standard Architecture; Laajennusväylästandardi, joka esiteltiin IBM PC:ssä vuonna 1981
Klusteri	kokoelma erityyppisiä signaaleja yhdistettynä yhdeksi signaaliksi
LED	Light-Emitting Diode eli hohtodiodi on puolijohdekomponentti, joka säteilee valoa kun sen läpi johdetaan sähkövirtaa.
Multiplekseri	MUX; elektroninen laite, jolla voidaan valita yksi useasta tulosignaalista ja ohjata se lähtöön.
NI-DAQ	National Instrumentsin valmistama tiedonkeruukortti
Stringi	merkkijono
Task	Ali-VI, jolla saadaan operoitua tiedonkeruukorttien kanavia diagrammissa
VI	virtuaali-instrumentti; LabVIEW-tiedostojen tiedostopääte sekä termi, jolla aliohjelmia sekä -funktioita kutsutaan.

Sisällys

1 Johdanto

Tämän insinöörityön tarkoituksena on toteuttaa Delcon Oy:lle puolijohdereleiden mittalaitteiston ohjelmallinen osuus LabVIEW-ohjelmointiympäristössä. Työn alussa käsitellään LabVIEW-ohjelmointiympäristön perusteita, jonka jälkeen seuraa lyhyt esittely testilaitteistosta. Tämän jälkeen käydään läpi testiohjelmassa käytetyt ratkaisut sekä kuvataan ohjelman toiminnallisuus. Työssä ei tarkastella LabVIEW'tä kriittisesti ohjelmointikielenä.

Aihepiiri sisältää useita englanninkielisiä termejä, joiden käyttö on vakiintunut myös suomen kielessä. Tällaisten termien kohdalla lienee järkevä esittää ne tekstin yhteydessä myös englanninkielisenä. Pelkät vapaat suomennokset saattaisivat johtaa virheellisiin tulkintoihin. Mittaus- sekä ohjelmistotekniikassa monien asioiden esittäminen englanniksi on paljon luontevampaa eikä suomenkielistä vastinetta aina edes ole olemassa.

Delcon Oy on vuonna 1975 perustettu elektroniikka-alan yritys, joka valmistaa pulssimuuntajatekniikkaan perustuvia puolijohdereleitä. Suunnittelu ja loppukokoonpano suoritetaan Nummelassa Veikkoinkorven teollisuusalueella. Puolijohdereleiden lisäksi Delcon Oy valmistaa ja myy erilaisia oheistuotteita, kuten asennuskantoja ja logiikka-adaptereita.

Projektin tavoitteena oli suunnitella ja toteuttaa puolijohdereleiden testilaitteistojen uusiminen. Testilaitteiston ISA-väyläiset tiedonkeruukortit olivat auttamattomasti vanhentuneita, joten uuteen testilaitteistoon oli hankittava uudet USB-liitäntäiset tiedonkeruukortit. USB-liitäntäiset tiedonkeruukortit varmistaisivat myös laitteiston siirrettävyyden työasemalta toiselle. Vanhaa testilaitteistoa vaivasi yleinen hitaus sekä mittausperiaatteesta johtuvat mittausvirheet. Uuden laitteiston oli määrä olla USB-liitäntäisten datankeruukorttien ansioista tarkempi ja nopeampi kuin nykyinen testilaitteisto. Uudella testerillä tulisi myös testata sekä tulo- että lähtöreleet, joihin nykyisellä kalustolla tarvitaan kaksi erillistä laitteistoa. Tuotekehityspäällikkö Reima

Konttisen kanssa aloitimme uuden laitteiston suunnittelemisen työntekijöiltä saadun palautteen perusteella.

2 LabVIEW

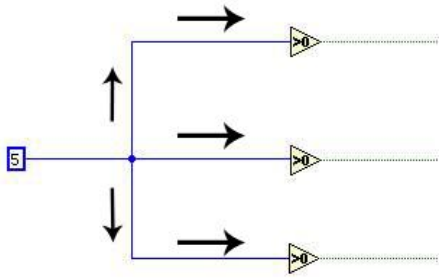
2.1 Historia

Vielä 1990-luvulla valtaosa tietokonepohjaisista mittaus- ja testausjärjestelmistä ohjelmoitiin perinteisillä ohjelmointikielillä, kuten C-kielellä ja Pascalilla. Nykyään suurin osa näistä sovelluksista kehitetään mittausjärjestelmien ohjelmointiin tarkoitetuilla sovelluskehittimillä, joista yleisin on National Instrumentsin LabVIEW.

National Instruments on yksi maailman suurimmista mittauskorttien valmistajista, jolla syntyi 1980-luvulla tarve hyödyntää tietokoneita mittaustulosten jatkokäsittelyssä. Tältä pohjalta syntyi kaupallinen sovelluskehitysympäristö LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench), jonka ensimmäisen version National Instruments esitteli MacIntosh-tietokoneelle vuonna 1986. LabVIEW'n ympärille on internetissä kasvanut laaja yhteisö, jossa käyttäjät kysyvät ja tarjoavat apua toinen toisilleen. [1.]

2.2 LabVIEW ohjelmointikielenä

LabVIEW on itse asiassa sovelluskehitin, jolla tuotetaan graafista koodia. Tällaista ohjelmakoodia kutsustaan usein G-koodiksi. LabVIEW poikkeaa perinteisistä ohjelmointikielistä myös suoritustapansa suhteen. Tavallista tekstipohjaista koodia suoritetaan rivi kerrallaan, kun taas LabVIEW'n graafisessa koodissa useita aliohjelmia voidaan suorittaa samanaikaisesti [kuva 1]. Tässä tapauksessa kokonaisluku viisi jaetaan kolmeen eri signaaliin, jotka tuodaan kolmen funktion tuloportteihin. Funktiot tarkistavat, sisältääkö signaali nollaa suuremman kokonaisluvun. Funktiot suoritetaan siis rinnakkain ja niistä lähtee ulostulosignaalina *Tosi*-tyyppinen muuttuja.



Kuva 1. G-kielisen koodin moniajo

LabVIEW'n graafisen koodin luonteesta johtuen aloituskynnys ohjelmointiin on matalampi kuin perinteisillä ohjelmointikielillä. Tämä on toisaalta G-kielisen koodin vahvuus, mutta toisaalta kokemattomalle ohjelmoijalle haaste, sillä graafisesta koodista saa aloittelija helposti tuotettua vaikeasti luettavaa ja sekavaa ”spagettikoodia”. Tästä syystä LabVIEW-ohjelmoinnissa on tärkeä muistaa hyviin ohjelmointitapoihin kuuluvat seikat, kuten funktioiden selkeä sijoittelu sekä selkeät, suorat johtimet. G-koodi on tämän lisäksi suunniteltu luettavaksi vasemmalta oikealle, joten ohjelmoijan on kaikin keinoin vältettävä johdottamasta oikealta vasemmalle. Objekteja ei myöskään ole syytä asetella johdotuksen päälle, koska tällöin sovelluksen luettavuus kärsii. [2, s. 5-10; s. 5-13.]

2.3 Virtuaali-instrumentit

LabVIEW-sovelluksia kutsutaan virtuaali-instrumenteiksi, koska ne muistuttavat reaalimaailman mittauslaitteita, kuten oskilloskooppeja tai yleismittareita. Virtuaali-instrumentti koostuu tietokoneesta ja siihen kytkettävistä lisälaitteista, joilla mitataan erilaisia reaalimaailman suureita. Ohjelmamoduulien nimen tiedostopäätteenä käytetään lyhennettä ”vi”, eli ohjelma voi olla nimeltään esimerkiksi *Mittaus.vi*. Jokaisessa vi:ssä on oltava etupaneeli eli käyttöliittymä sekä G-koodista koostuva osuus, jota kutsutaan diagrammiksi. LabVIEW'ssä on käytettävissä laaja valikoima valmiita funktioita. Datan keruuta varten LabVIEW'ssä on myös valmiit ali-vi:t, joilla pystytään helposti ohjaamaan lisälaitteita. [2, s. 2-1.]

Usein on tarpeellista tallentaa saatuja mittaustuloksia helposti luettavaan muotoon. Tämän vuoksi LabVIEW'ssä on monia taulukonkäsittelyfunktioita valmiiksi sisäänrakennettuna. Tallennuksessa käytetään yleisesti sarkaimin erotettua taulukkomuotoa, jota pystytään lukemaan kaikilla taulukkolaskentaohjelmilla käyttöjärjestelmästä riippumatta.

Yksi LabVIEW-ohjelmointiympäristön vahvuuksista on ominaisuus, jota kutsutaan virtuaali-instrumenttien ketjuttamiseksi. Tällä tarkoitetaan aliohjelmien yhdistämistä ketjuksi, jossa ainoana vaatimuksena on tulo- ja lähtösignaalien yhteensopivuus. Näin pääohjelma saadaan selkeäksi sekä helposti luettavaksi.

2.4 Datankeruu

LabVIEW-sovellukseen voidaan tuoda dataa eri muodoissa tiedonkeruukorttien (DAQ) avulla [kuva 2]. Tiedonkeruukorteilla voidaan mitata sekä sähköisiä että fysikaalisia ilmiöitä, kuten jännitettä, virtaa, lämpötilaa, painetta tai ääntä. Useissa tiedonkeruu-korteissa on myös analogialähtöjä, kuten +5 voltin käyttöjännitteen syöttö, sekä erinäisiä digitaalilähtöjä.

Kuva 2. National Instrumentsin DAQ [3.]

DAQ:t muuntavat analogiset signaalit näytteistämällä tietokoneen ymmärtämiksi digitaalisiksi lukuarvoiksi, jotta niitä voidaan jatkokäsitellä LabVIEW'llä. Perustason tiedonkeruukorteissa on kustannussyistä ainoastaan yksi A/D-muunnin, joka lukee useaa kanavaa limittäjän (eng. multiplexer) avulla. [3.]

3 Testilaitteisto

3.1 Vaatimukset

Tuotteet testataan Delcon Oy:ssä kahteen kertaan, komponenttien käsiladonnan jälkeen piirilevyvaiheessa sekä lopputuotevaiheessa koteloituna [5]. Näiden testien välissä suoritetaan myös 4,3 kilovoltin läpilyöntikoestus. Piirilevyvaiheessa tuotteet testataan neljän kappaleen aihioissa, loppuvaiheessa yksittäin [8]. Tästä syystä testipenkissä tulisi olla molempiin vaiheisiin sopivat adapterit. Testaus tapahtuu kannellisessa laatikossa, koska tuotteen LED-ilmaisimen testaamiseen vaaditaan pimeä tila. Laatikon kannessa tulee olla kytkin, joka estää testauksen käynnistämisen ellei kansi ole huolellisesti suljettu. Käyttöliittymän osalta tärkein vaatimus on helppokäyttöisyys. Testaaminen on saatava mahdollisimman sujuvaksi tietotekniikan perustaidot omaavaa peruskäyttäjää ajatellen. Tästä johtuen etupaneelista tulee karsia kaikki yksityiskohtainen tieto pois peruskäyttäjän näkymästä.

3.2 Delcon-rele

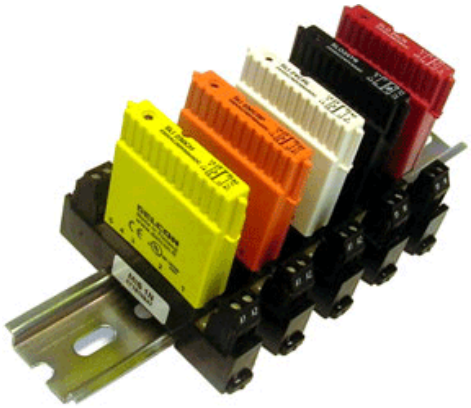
Välireleiden äkillinen rikkoutuminen aiheuttaa teollisuudessa useasti kalliiksi käyviä tuotantokatkoksia. Tämän vuoksi releitä vaihdetaan ennakoivasti osana normaalia kunnossapitoa, jotta yllättäviltä katkoksilta vältyttäisiin. Releiden toimintaan ja elinikään vaikuttavat useat tekijät, kuten kytkentätaajuus, erilaiset kuormat ja teollisuusympäristössä esiintyvät häiriöt. Nämä tekijät yhdessä lyhentävät välireleiden käyttöikää ratkaisevasti [6].

Sähkömekaanisen releen elinajaksi annetaan yleensä noin kymmenen miljoonaa kytkentää. Tämä on kuitenkin kuormittamaton arvo, joten todellisuudessa kolmen ampeerin kuormalla releen elinikä voi olla noin miljoonan kytkennän luokkaa. Käytännössä tämä tarkoittaa sitä, että jatkuvasti käynnissä olevissa laitoksissa rele

kestää arviolta vuodesta kahteen. Delcon-rele on immuuni sekä signaali- että kuormapuolen häiriöille, joten se soveltuu mainiosti vaativiin teollisuusolosuhteisiin. Delcon-releen odotettu elinikä on noin 15 vuotta, mikä vastaa normaalin automaatiolaitteiston käyttöikää.

Delcon-releiden toiminta perustuu pulssimuuntajatekniikkaan. Kytkentäpiiri saa tarvitsemansa energian signaalista muuntajan kautta, toisin kuin optoerottimilla varustetut puolijohdereleet. Tällä periaatteella pystytään eliminoimaan suurin osa perinteisten puolijohdereleiden heikkouksista teollisuusolosuhteissa. Lisäksi releen päälle- ja poiskytkentää ohjaavat hystereesipiirit, joten päällekytkeytyminen tapahtuu huomattavasti suuremmalla käyttöjännitteellä kuin poiskytkeytyminen. Tällä varmistetaan, että kytkentä on aina puhdas ja nopea. Tilaa ilmaiseva LED syttyy ja sammuu vastaavasti, joten myös sen näyttö on aina sataprosenttisesti luotettava.

Delcon-releet jaetaan kahteen eri luokkaan, tulo- ja lähtöreleisiin. Tuloreleillä muunnetaan kentältä saapuva signaali tasajännitteeksi. Tuloreleiden tulojännite voi olla AC- tai DC-muotoista, mutta lähtöjännite on aina 0...60 VDC (tyypillisesti 24 VDC). Lähtöreleitä käytetään tyypillisesti ohjaamaan pienellä ohjaussignaalilla isoja kuormia. Lähtöreleitä valmistetaan sekä AC- että DC-ohjauksella, ja kuormajännite voi olla 0…350 VDC tai 0...415 VAC. Releitä valmistetaan kahta eri tuoresarjaa: DIN-kiskoasennukseen sopivaa SL-sarjaa [kuva 3] sekä piirilevyasennukseen sopivaa GL-sarjaa [kuva 4].

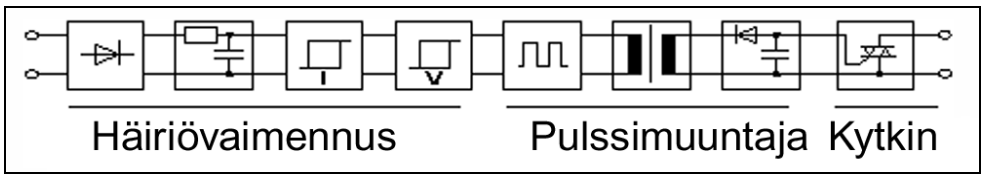


Kuva 3. SL-sarjan Delcon-releet asennuskantoineen

Kuva 4. GL-sarjan puolijohdereleitä

Releen lohkokaavio

Kaikissa Delcon-releissä käytetään useita häiriönpoistopiirejä, joilla eliminoidaan signaalipuolelta tulevat häiriöt. Tällä varmistetaan puhtaan signaalin siirtyminen muuntajan läpi kytkentäpiirille [kuva 5]. Pulssimuuntajalla toteutetaan 4300 VAC tai 4600 VAC galvaaninen erotus ohjauspuolen ja kentän välillä. Se välittää signaalin ja antaa kytkentäpiirille sen tarvitseman tehon. Kytkentäkomponenttina käytetään releestä riippuen joko triakkia tai teho-MOSFET:teja. [4.]



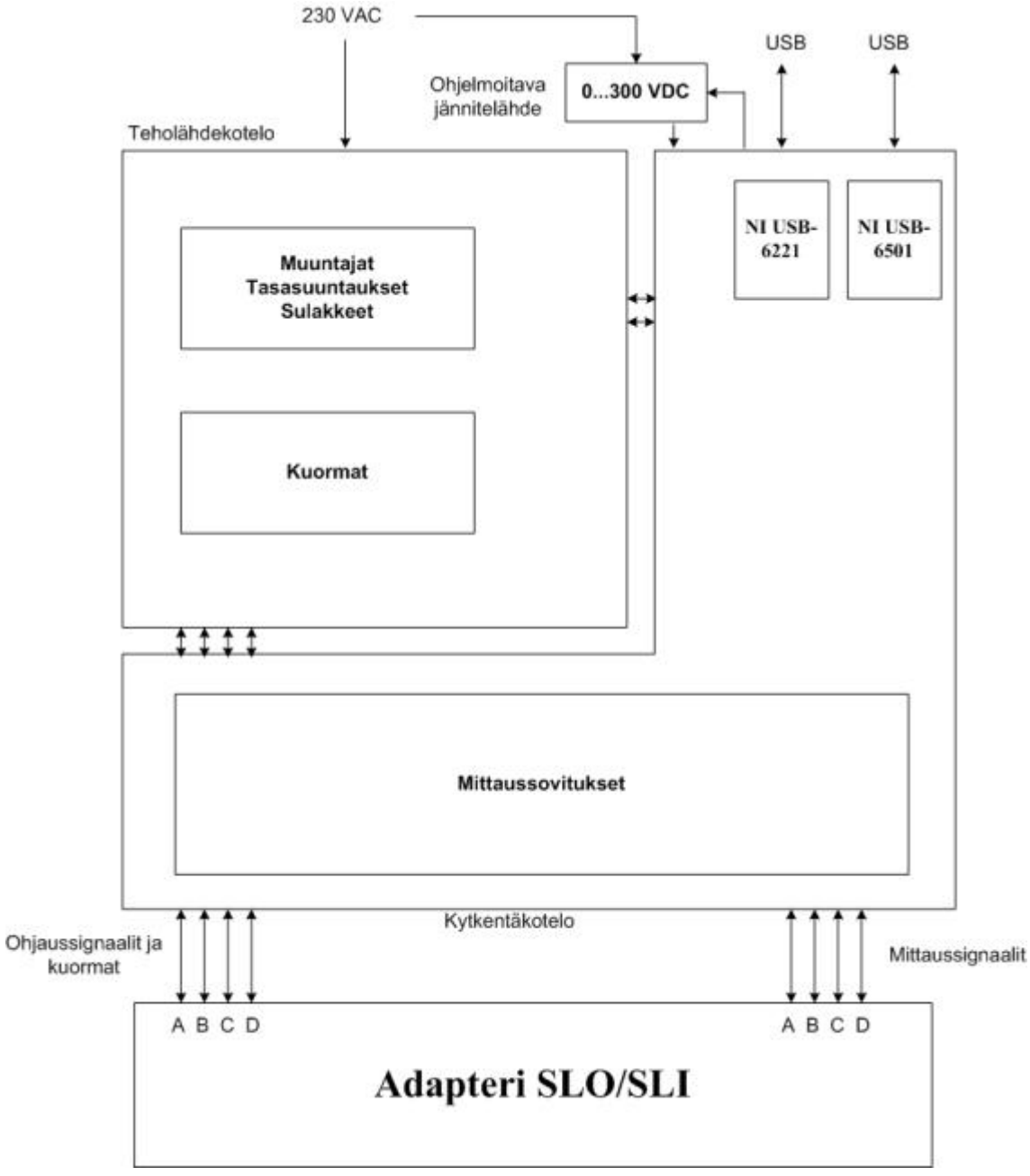
Kuva 5. Delcon-releen lohkokaavio

Tavallisessa AC-lähtöreleessä käytetään huomattavasti vahvempaa triakkia verrattuna optoerotettuihin puolijohdereleisiin, mikä mahdollistaa selvästi korkeamman kuorman jännitteen muutoksen $\frac{dV}{dt}$. Yleisesti puolijohdereleiden suojana toimiva RC-piiri voidaan jättää pois ja korvata se varistorilla, joka toimii myös tehokkaana transienttisuojana.

Tällaisen lähtöreleen vuotovirta on hyvin pieni (< 0,05 mA), ja rele on immuuni teollisuusympäristössä yleisille kuorman puolelta tuleville häiriöpiikeille. Triakkireleen päästöviive vaihtelee tyypillisesti 50 hertsin sähköverkon luonteesta johtuen yhdestä kymmeneen millisekuntiin, joten nopeaan AC-ohjaukseen tarvitaan MOSFET:lla varustettu rele. Näillä releillä päästöviive on tyypillisesti 0,3 millisekuntia. [7.]

3.3 Yleiskatsaus

Testilaitteisto koostuu kolmesta lohkosta: adapterista, teholähde- sekä kytkentäosiosta [kuva 7] Kytkentä- sekä teholähdekotelosta on liitetty kuvat työn liiteosioon. Lisäksi käytetään ulkoista jännitelähdettä, josta saadaan 0…5 VDC:n ohjauksella tuotettua 0…300 VDC:n lähtöjännite.



Kuva 6. Testilaitteiston lohkokaavio

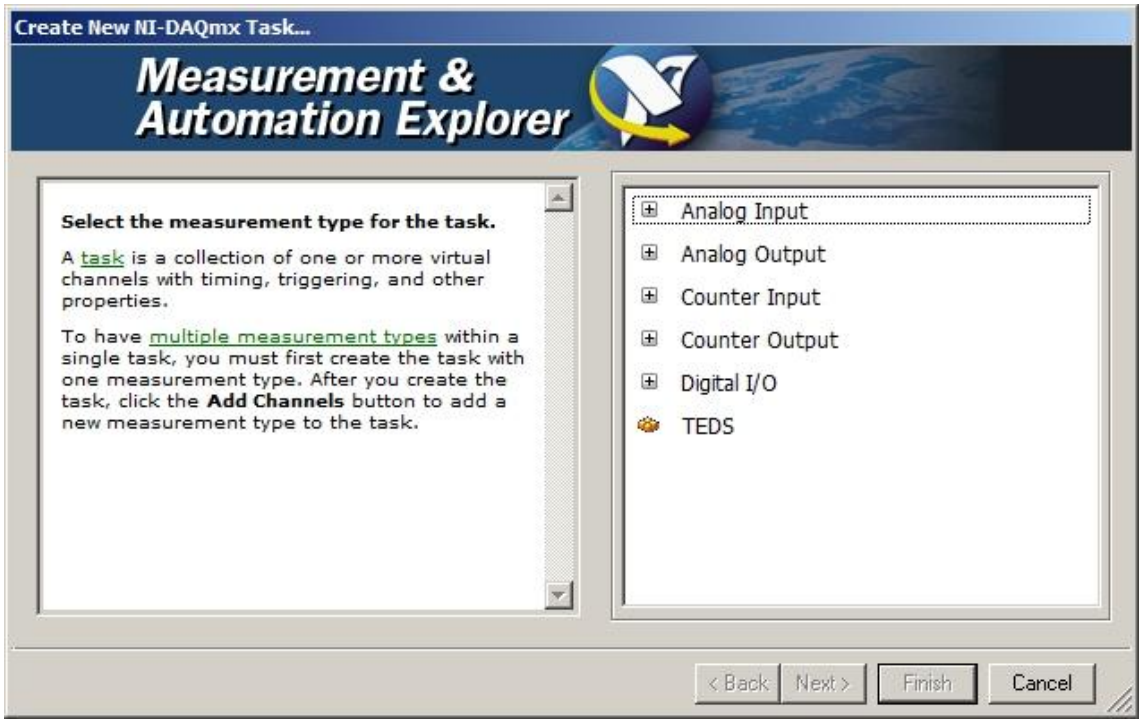
Mittauskorteiksi on valittu digitaali-I/O-kortti (NI USB-6501) sekä multi-I/O-kortti (NI USB-6221), jotka on liitetty tietokoneeseen USB-väylällä.

Adapterikotelo on varustettu testipiikein, joiden avulla saadaan kontakti testattaviin releisiin. Tuotteen LED:n tilan mittaamista varten adapterissa on neljä kappaletta optisia sensoreita, jotka vaativat pimeän tilan mittaustapahtuman ajaksi. Kotelo on tästä syystä varustettu suljettavalla kannella, jotta mittausta häiritsevän valovuodon määrä olisi mahdollisimman pieni. Adapterin kanteen vasempaan reunaan on asennettu kytkin, jotta testirutiini saadaan käynnistymään käyttäjän sulkiessa kotelon luukun.

3.4 Testilaitteiston kanavamäärittely

Käytettävät kanavat määritellään Measurement & Automation Explorer (MAX) -ohjelmassa tehtäviin (eng. task). Yhteen tehtävään voidaan määritellä monta fyysistä kanavaa, jolloin sitä kutsuttaessa mitataan kaikki kyseisen nipun kanavat [2, s.1-4]. Tehtävän luonti tapahtuu seuraavasti:

1. Valitaan työkaluriviltä Create New NI-DAQmx Task…
2. Seuraavaksi valitaan kanavan tyyppi: analogia/digitaali, lähtö/tulo [kuva 9].



Kuva 7. Uuden tehtävän luonti

3. Tämän jälkeen valitaan mitattava tai syötettävä suure.
4. Lopuksi ohjelma listaa määrityksiä tukevat fyysiset kanavat listaan, josta käyttäjä voi valita haluamansa kanavan. NI USB-6501 sisältää ainoastaan digitaalisia I/O-kanavia, kun taas NI USB-6221 on multi-I/O-tyyppinen mittauskortti, jossa on sekä digitaali- että analogiakanavia.

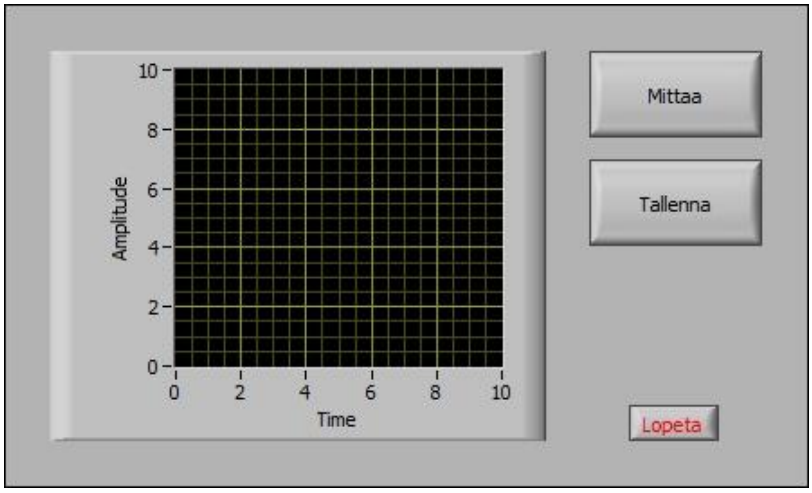
Samaan tapaan syötetään jokainen käytössä oleva kanava. Projektissa käytetty kanavaluettelo löytyy kuvauksineen liiteosiosta [liite 1]. Tehtävän ominaisuuksista voidaan tämän jälkeen muuttaa kanaviin liittyviä ominaisuuksia, kuten mitattavaa suuretta, näytteistystaajuutta ja mittausaluetta. Onnistuneen mittaustuloksen kannalta on syytä ottaa huomioon *Terminal Configuration* -asetus, jolla määritetään potentiaali, jonka suhteen mittaus suoritetaan.

4 Sovellus

4.1 Yleiskatsaus

Pääohjelman *VI* on toteutettu rakenteella, jonka nimi on *Event Handler*. Ohjelma pyörii *while*-silmukassa ja suorittaa osia *VI*:stä vain silloin kun tapahtumalle vastaavaa kytkintä painetaan etulevystä. Tapahtumaksi voidaan määritellä myös jokin ulkoinen signaali tai ohjelmallinen osa sovellusta. Jokaista tapahtumaa kohden voidaan määrätä etulevystä kytkimet, johon kyseinen tapahtuma reagoi. *Event Handler* -rakenteen taustalla pyörii *while*-silmukka, joka pitää *VI*:n käynnissä. Tämä silmukka keskeytetään etulevyn *Lopeta*-painikkeella, jolloin koko ohjelman ajaminen lopetetaan. Ratkaisun etuna on *VI*:n reagointi käyttäjän antamiin komentoihin tehokkaammin ja nopeammin kuin perinteinen toistuva tarkkailu (eng. polling). Prosessoritehon säästämisen lisäksi *Event Handler* -rakenne pitää huolen siitä, että *diagrammi* rekisteröi etulevyn painallukset oikeassa järjestyksessä. [2, s. 8-15.]

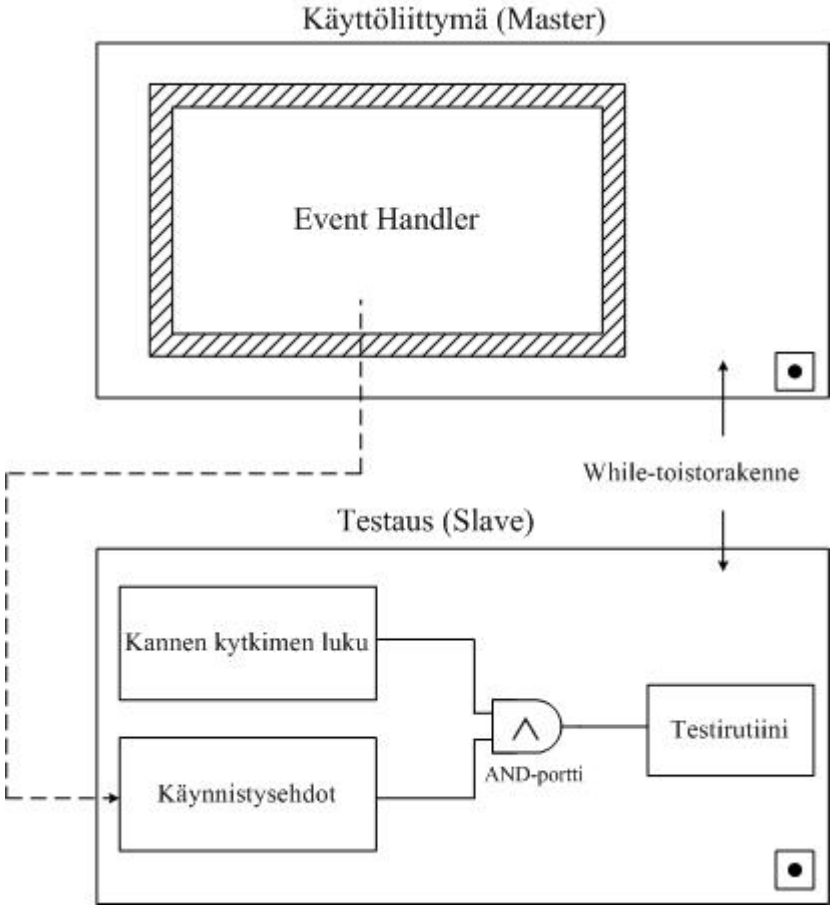
Seuraavassa esimerkissä tarkastellaan ongelmaa, joka esiintyy, kun *Event Handler* -rakennetta ei käytetä. Tutkitaan yksinkertaista etulevymallia, joka muodostuu kolmesta painikkeesta: *Mittaa, Tallenna* sekä *Lopeta* [kuva 10]:



Kuva 8. Event Handler -esimerkki

Oletetaan, että käyttäjä painaa kahta ensimmäistä painiketta nopeasti peräkkäin. *Diagrammin* rinnakkaisesta suoritusjärjestyksestä johtuen *VI* tulkitsee komennot joko järjestyksessä *Tallenna-Mittaa*, tai *Mittaa-Tallenna*, jotka johtavat täysin erilaiseen lopputulokseen. Käytännössä käyttäjä ei kuitenkaan ehdi painamaan painikkeita näin nopeasti, vaan LabVIEW rekisteröi painallukset usein oikeassa järjestyksessä ilman *Event Handler* -rakennetta. *Event Handler* -rakenteella saadaan kuitenkin varmistettua, että LabVIEW käsittelee painikkeille määrätyt toiminnot jono-tyyppisesti ja jokainen etulevyn painallus rekisteröidään varmasti oikeassa järjestyksessä.

Event Handler -rakenteen lisäksi pääohjelmassa on käytetty *Master & Slave* -rakennetta [kuva 11]: *Master*-osiossa pidetään yllä käyttöliittymää ja kerätään käyttäjältä tietoja sitä mukaa kuin niitä syötetään.



Kuva 9. Master & Slave -rakenne

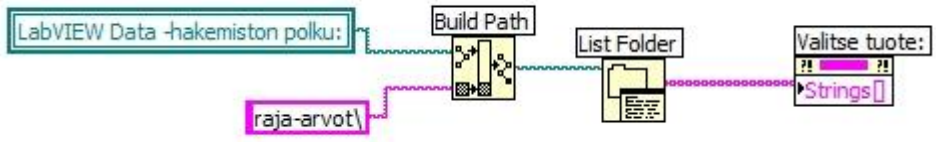
Event Handler -rakenteen luonteesta johtuen käyttöliittymän *while*-toistorakennetta ei suoriteta, ellei käyttäjä anna käskyä eli herätettä etulevyltä. Näin säästetään tietokoneen resursseja ohjelman jäädessä tausta-ajolle. *Slave*-osiota suoritetaan rinnakkain käyttöliittymän ohella: Mikäli käyttöliittymäosiosta luettavat käynnistysehdot eivät täyty, *Slave*-osiossa ainoastaan luetaan kannen kytkimen tilaa. Mikäli käynnistysehdot on oikein syötetty, *AND*-portin molemmat ehdot täyttyvät ja testirutiini käynnistyy.

4.2 Alustukset ja ohjelman rakenne

Alustukset suoritetaan sekvenssirakenteen ensimmäisessä osassa, jonka jälkeen siirrytään ohjelman toiseen osaan, jossa ydinosa pyörii *Event Handler* -rakenteen avulla.

Alustusosiossa nollataan kaikki *VI*:n käyttämät muuttujat ja tiedonkeruukorttien digitaalilähdöt, poistetaan vilkkumisefektit tuotekohtaisista kentistä sekä tutkitaan raja-arvohakemistoa uusien tuotteiden varalta.

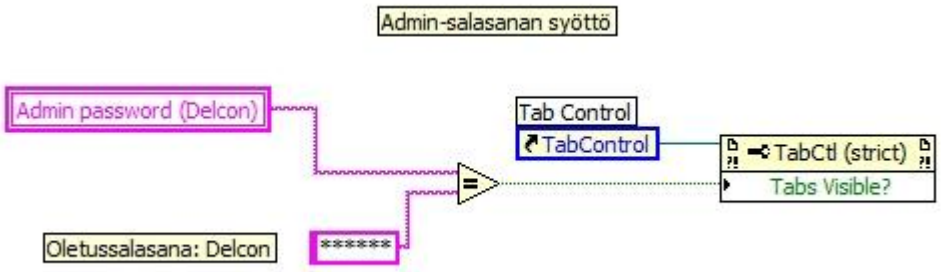
Mikäli halutaan testata uutta tuotetta, riittää uuden raja-arvotaulukon luonti raja-arvo -hakemistoon. Ohjelma tutkii alustusosiossa raja-arvohakemiston lisäten uudet tuotteet automaattisesti pudotusvalikkoon [kuva 12].



Kuva 10. Raja-arvotaulukoiden luku

Admin-välilehti

Admin-välilehti on piilotettu peruskäyttäjältä asettamalla etulevyyn ylläpidon määrittämä salasana, jonka syöttämällä välilehti ilmestyy näkyviin. Saadakseen *Admin*-välilehden näkyviin käyttäjän tulee syöttää oikea tunnus salasanakenttään ja painaa *Enter*-painiketta. Salasanan tarkistuksessa käyttäjän syöttämää merkkijonoa verrataan ennalta-asetettuun merkkijonoon, ja jos merkkijonot ovat yhtäläiset, välilehdet asetetaan näkyviin muuttamalla välilehtien näkyvyyttä *Property-node* -toiminnolla [kuva 13].

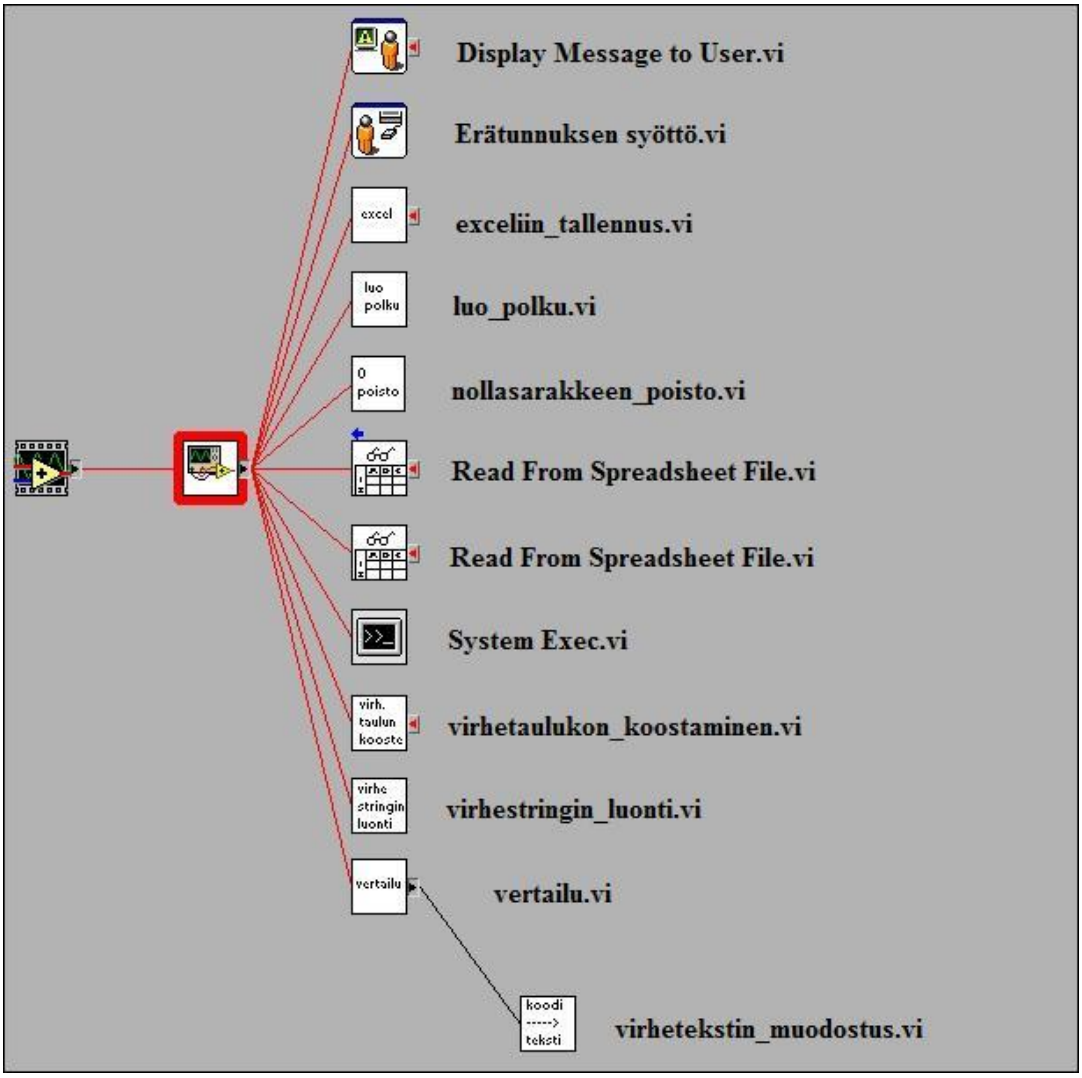


Kuva 11. Admin-salasanan tarkastus

Tämä tarkistus suoritetaan *VI*:n alustusten yhteydessä sekä *Event Handler* -rakenteen avulla aina kun käyttäjä muuttaa etulevyn salasanakenttää. *Admin*-välilehdeltä voidaan määrittää käytettävän taulukkolaskentaohjelman, raja-arvo- sekä tallennushakemiston polut. Tällä tavoin ohjelman siirrettävyys tietokoneelta toiselle on pyritty tekemään mahdollisimman vaivattomaksi. Myös tuotekehitysosastoa varten tarkoitettu peruskäyttäjälle epäoleellinen tieto tulostetaan tänne.

4.3 VI-hierarchy

Testerisovellus koostuu pääohjelmasta, jonka etulevy toimii sovelluksen käyttöliittymänä sekä useista alemman tason virtuaali-instrumenteista [kuva 14]:



Kuva 12. VI-hierarchy

Display Message to User.vi

Display Message to User.vi on eräs *VI*-kirjaston apufunktioista, jolla saadaan tulostettua käyttäjälle viesti ponnahdusikkunassa. Ratkaisun etuna on käyttäjän huomion saaminen täsmällisesti verrattuna perinteisiin tekstikenttiin. Virtuaali-instrumenttiin voidaan tuoda parametreinä loogis-tyyppinen signaali, jolloin muotoa ”tosi” oleva signaali aiheuttaa viestin ponnahtamisen käyttäjälle. Ikkuna voi olla pelkkä ilmoitus tai siihen voidaan määrittää erilaisia painikkeita, kuten ”hyväksy” ja ”peruuta”. Painikkeiden signaaleja voidaan tämän jälkeen välittää virtuaali-instrumentilta eteenpäin ja niitä voidaan jälleen käyttää luomaan haluttua toiminnallisuutta.

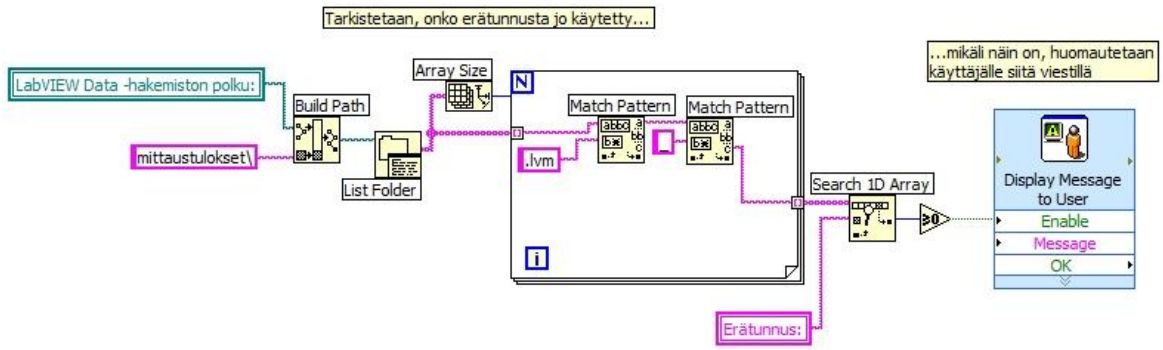
Erätunnuksen syöttö.vi

Jos käyttäjä syöttää jo olemassa olevan erätunnuksen, ohjelma varoittaa siitä [kuva 15]. Erätunnuksien vertailu aloitetaan listaamalla mittaustulosten tallennuspaikkana oleva hakemisto: *List Folder.vi* palauttaa hakemiston kaikki tiedostot yksiulotteisena

taulukkona, joka on muotoa $1 \times n$, jossa *n* = tiedostojen lukumäärä. *Array Size.vi*

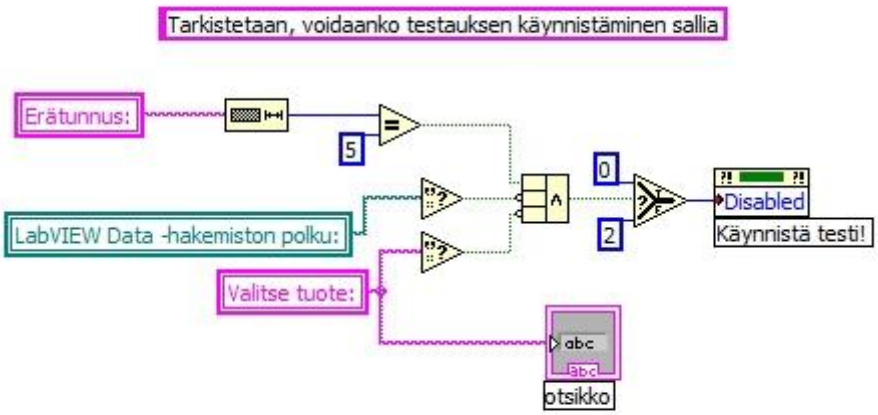
-funktio palauttaa (yksiulotteisen taulukon tapauksessa) kokonaislukuna taulukon koon *n*, joka syötetään laskuriksi *For*-toistorakenteeseen. Näin saadaan toistorakenteen toiminnallisuus toistettua taulukon jokaiselle alkiolle.

Toistorakenteen sisällä etsitään tiedostonimestä erätunnus *Match Pattern.vi* -funktiota apuna käyttäen. Indeksointitoiminnon ansioista toistorakenteesta lähtevä signaali on jälleen yksiulotteisen taulukon muodossa. Mittaustulokset tallennetaan muotoon *tuote_erätunnus.lvm*, joten oleellinen informaatio saadaan leikkaamalla ensin tiedostopääte *.lvm* pois, jonka jälkeen seuraava VI ottaa talteen alaviivan jälkeisen merkkijonon (erätunnus). Tämän jälkeen taulukossa on jäljellä ainoastaan erätunnukset, joista voidaan etsiä käyttäjän syöttämä erätunnus *Search 1D Array.VI* -funktion avulla.



Kuva 13. Erätunnuksen tarkistus

Jotta testirutiinin käynnistäminen voidaan sallia, seuraavien ehtojen on toteuduttava: erätunnuksen on oltava viisi tai kuusi merkkiä pitkä ja käytettävä työhakemisto ja testattava tuote on määritelty oikein. Mikäli edellä mainitut tiedot on syötetty onnistuneesti, testirutiinin käynnistys sallitaan ja *Käynnistä*-painike vapautuu lukituksestaan [kuva 16].



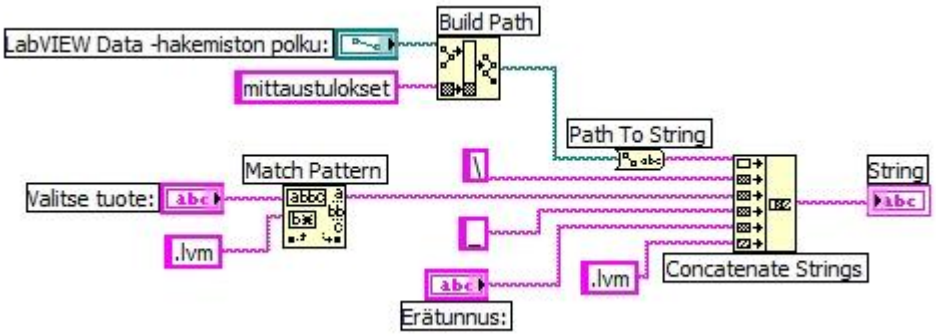
Kuva 14. Testirutiinin käynnistämisen vaatimukset

Taulukkoon tallennus.vi

Tallennus tapahtuu *Write to Spreadsheet File.vi* -funktiota käyttäen *exceliin_tallennus.vi* -aliohjelmassa. Tallennusprosessia käsitellään tarkemmin luvussa 4.5.

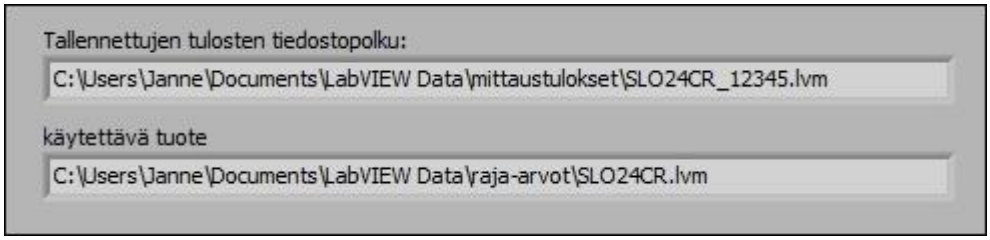
Luo polku.vi

Ennen tallennustapahtumaa on tarpeen koostaa absoluuttinen polku tarvittavalle tallennustiedostolle [kuva 17].



Kuva 15. luo_polku.vi

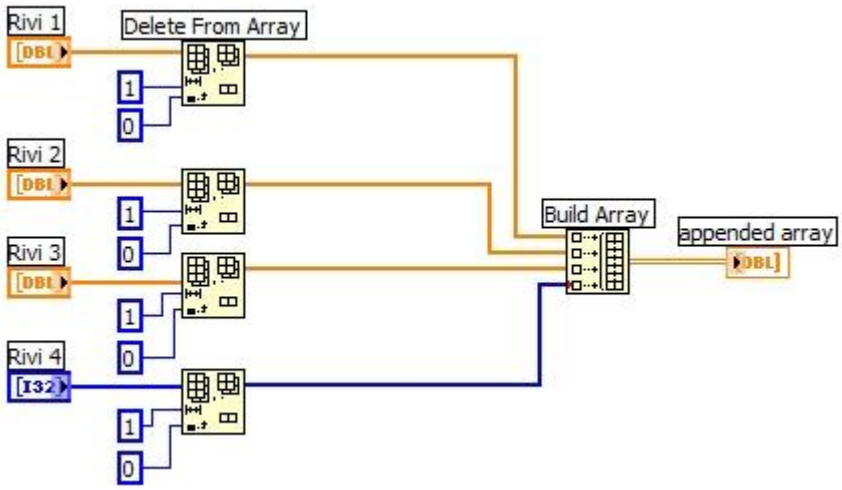
Koostaminen aloitetaan luomalla alihakemisto *mittaustulokset* käyttäjän määrittämän *LabVIEW Data* -hakemiston alle *Build Path.vi* -funktion avulla. *Valitse tuote* -muuttuja sisältää käyttäjän valitseman raja-arvotaulukon nimen, josta leikataan tiedostopääte *.lvm* pois. Näin jäljelle jää pelkkä tuotteen nimi, josta muodostetaan yhdessä edellä määritetyn hakemiston kanssa lopullinen polku merkkijonoksi *Concatenate Strings.vi* -funktion avulla [kuva 18].



Kuva 16. Tiedostopolut

Nollasarakkeen poisto.vi

Kymmenportaisen testirutiinin portaat on numeroitu yhdestä kymmeneen, mutta *While*-toistorakenteen laskurista johtuen ensimmäinen kierros ei ole numero yksi. Laskuri on numeroitu LabVIEW-ohjelmointiympäristössä siten, että nollakierros on ensimmäinen suoritettava kierros. Toistorakenne luo siis indeksoimalla taulukon, jossa on 11 saraketta. Ensimmäisellä kierroksella ei suoriteta testausta ollenkaan, joten ensimmäinen sarake on poistettava [kuva 19].



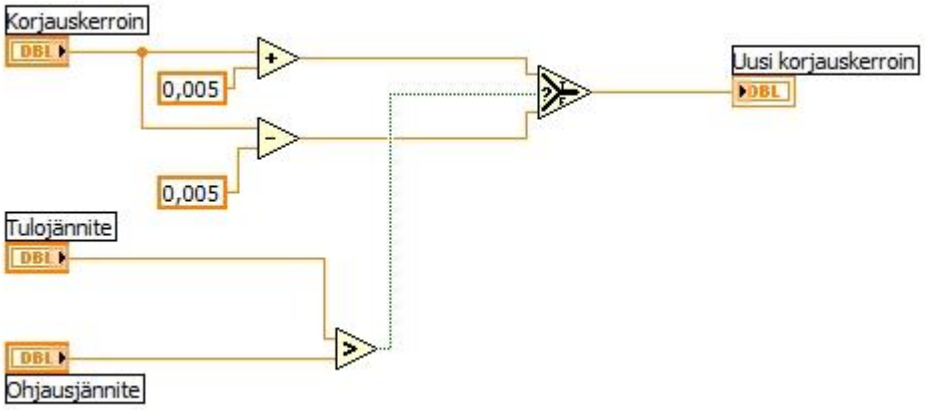
Kuva 17. Nollasarakkeen poisto

Signaalit tuodaan virtuaali-instrumenttiin yksiulotteisissa taulukoissa, jotka ovat muotoa 1×11 alkiota. *Delete From Array.vi* -funktiot poistavat taulukoista ensimmäiset alkiot: ensimmäisenä parametrinä funktioon syötetään kokonaisluku 1, joka kertoo funktiolle poistettavan osion pituuden. Toisena parametrinä syötetään kokonaisluku 0, joka viittaa taulukon ensimmäiseen alkioon. Näin saadaan koottua tallennusosiota varten 4×10 alkiota sisältävä taulukko, jossa sarakkeiden lukumäärä vastaa testien numerointia. Näin saatu taulukko muunnetaan vielä juuri ennen tallennusta lopulliseen yksiulotteiseen tallennusmuotoon 1×40 alkiota.

Lue taulukosta.vi

Read From Spreadsheet File.vi -funktio on sisällytetty *VI*-kirjastoihin, jotta taulukosta luku onnistuisi yhdellä funktiolla. Parametreinä voidaan määritellä muun muassa luettavan taulukon muoto, tiedostopolku ja rivien maksimimäärä.

Mikäli relettä ohjataan DC-jännitteellä, tuotteeseen syötetään suoraan taulukosta luettu jännitearvo. AC-jännitettä syötettäessä joudutaan käyttämään korjauskerrointa tulojännitteen korjaamiseksi vastaamaan ohjausjännitettä [kuva 20]:

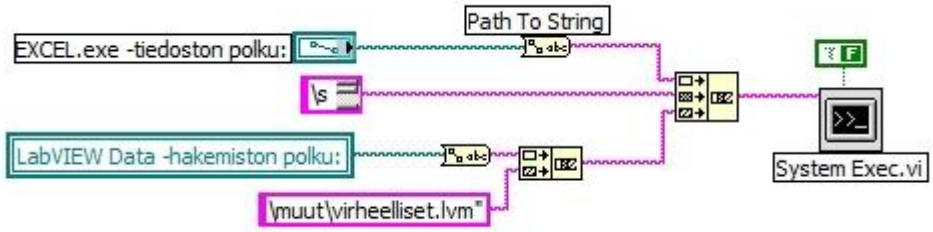


Kuva 18. Korjauskertoimen hienosäätäminen

Ohjausjännite luetaan suoraan raja-arvotaulukosta, kun taas tulojännite mitataan tuotteen navoista. Korjauskerroin tulee säätää niin, että jännitteiden ero olisi mahdollisimman pieni. Korjauskerrointa käytetään ohjausjännitteen jakajana ennen jännitteen syöttämistä releeseen. Sovellusta käynnistettäessä korjauskertoimelle annetaan alustusarvo, joka on lähellä kertoimen optimiarvoa. Aliohjelma tutkii jännitteiden eroja, ja mikäli tulojännite on ohjausjännitettä suurempi, lisätään vanhan korjauskertoimen arvoon vakio 0,005. Muussa tapauksessa korjauskertoimesta vähennetään sama vakioarvo. Tämän jälkeen uusi korjauskertoimen arvo tallennetaan vanhan kertoimen päälle. Kyseinen hienosäätö suoritetaan testeissä 2…8. Kyseistä korjauskertoimen käytetään seuraavan testirutiinin alustusarvona. Korjauskertoimen oletusarvo palautetaan alkuarvoonsa vasta kun koko sovellus käynnistetään uudelleen.

System Exec.vi

System Exec.vi -funktiota voidaan käyttää kuten Windows-käyttöjärjestelmän komentorivikehotetta. Syöttämällä merkkijonona halutun taulukkolaskentaohjelman *exe*-tiedoston ja perään lainausmerkeissä kohdetiedoston nimen avautuu tiedosto uuteen ikkunaan kyseisessä taulukkolaskentaohjelmassa pääohjelman pysyessä tausta-ajossa [kuva 21].



Kuva 19. Datan avaus taulukkolaskentaohjelmaan

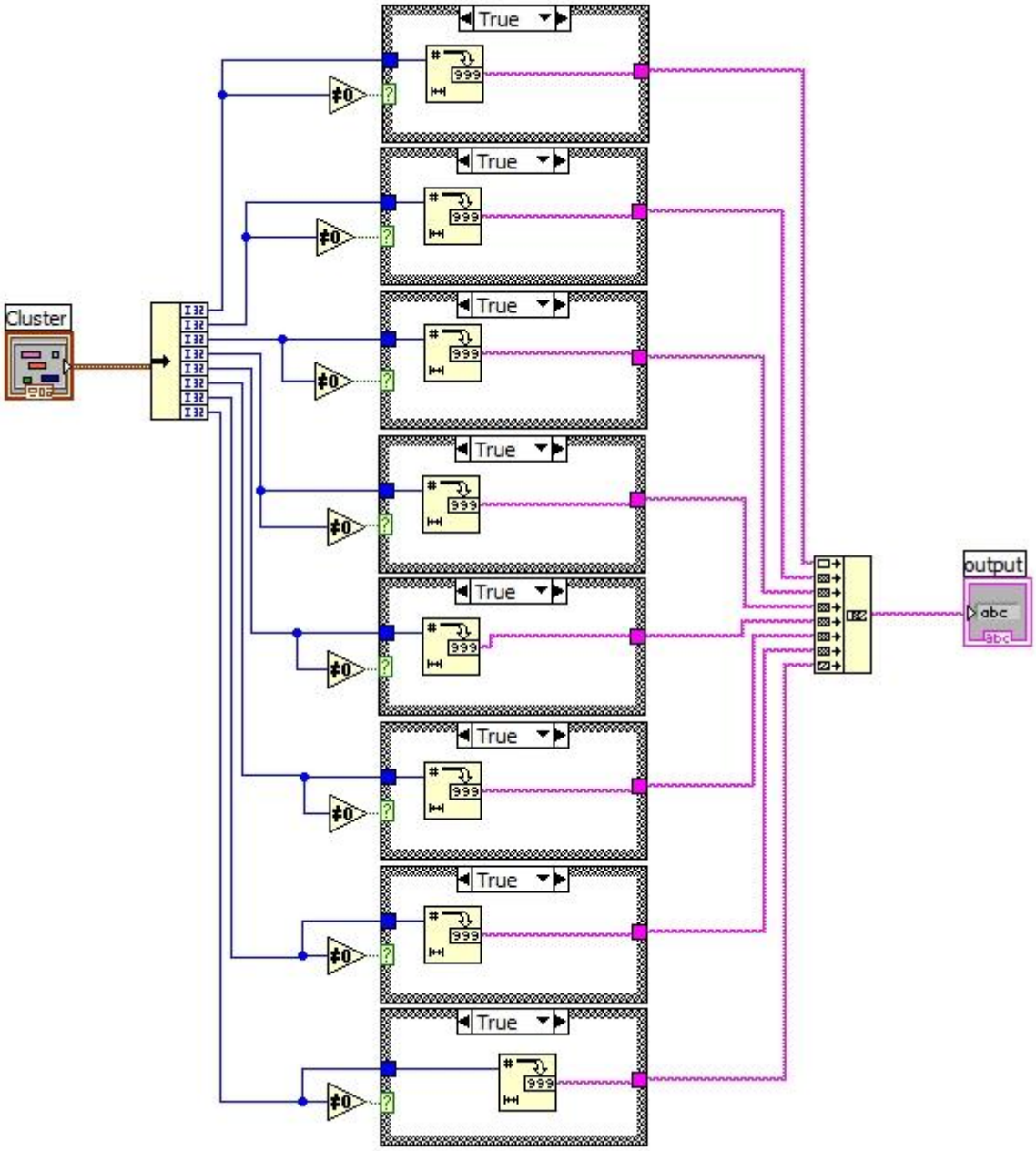
Pääohjelman normaali toiminta taustalla varmistetaan syöttämällä *System Exec.vi* -funktiolle parametrinä arvoltaan ”epätosi” oleva totuusarvomuuttuja. Näin pääohjelma ei odota taulukkolaskentaohjelman sulkemista, vaan jatkaa ohjelman suorittamista taustalla.

Virhetaulukon koostaminen.vi

Virhetaulukon koostaminen kuuluu osana tallennustapahtumaa, jota käsitellään tarkemmin luvussa 4.5.

Virhestringin luonti.vi

Virhekoodit tuodaan *Virhestringin luonti.vi* -aliohjelmaan kahdeksan kokonaislukua sisältävässä *cluster*-signaalissa. Jokaista kokonaislukua tutkitaan vuorotellen ja verrataan lukuarvoon nolla. Mikäli tunnus sisältää nollasta poikkeavan arvon, muutetaan tämä kokonaisluku merkkijonomuotoon *Number to String.vi* -funktiolla ja kootaan lähtösignaaliksi [kuva 22].

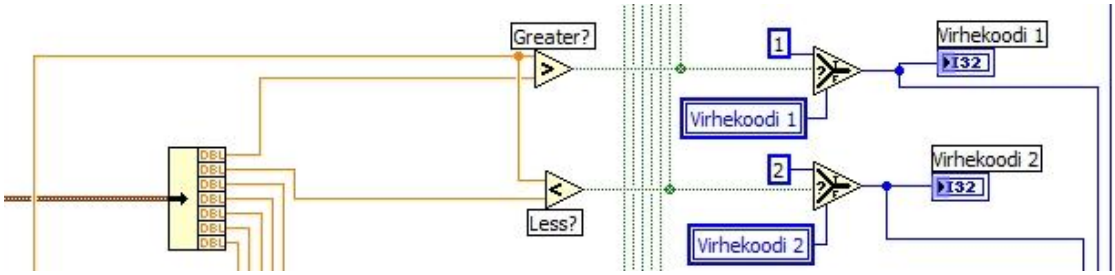


Kuva 20. Virhestringin luonti

Näin saatu numeroarvoista koostuva merkkijono tallennetaan virhelokiin, jota käsitellään tarkemmin luvussa 4.5.2.

Vertailu.vi

Testiprosessin ydin on saatujen mittaustulosten vertailu ennalta asetettuihin raja-arvoihin. Nämä raja-arvot luetaan taulukosta *Read from Spreadsheet File.vi* -funktiolla, jonka jälkeen ne tuodaan *cluster*-signaalissa *Vertailu.vi* -aliohjelmaan [kuva 23].

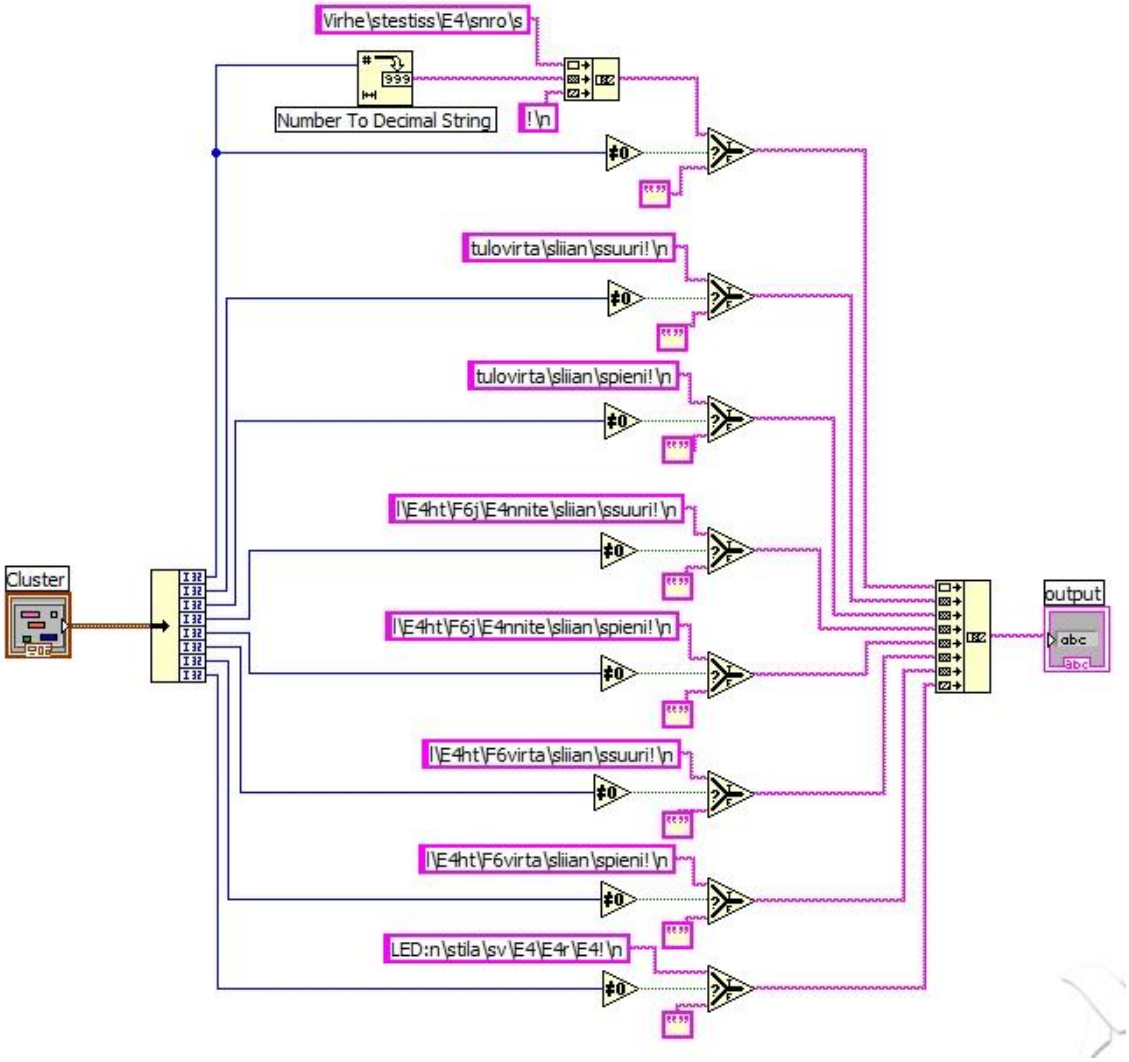


Kuva 21. Mitattujen arvojen vertailu raja-arvoihin

Vertailu.vi -aliohjelmassa raja-arvoja sisältävä *cluster*-signaali puretaan yksittäisiksi lukuarvoiksi. Tämän jälkeen mitatulle arvolle suoritetaan kaksi vertailua, joissa tutkitaan, onko mitattu arvo asetettua ylärajaa suurempi tai alarajaa pienempi. Tieto tapahtuneesta virheestä otetaan talteen, mikäli toinen ehdoista täyttyy. Samalla tallennetaan virhekoodimuuttujiin tieto siitä, missä vertailutapahtumassa virhe havaittiin.

Virhetekstin muodostus.vi

Virhetekstin muodostus.vi -aliohjelmaan tuodaan tulosignaalina kahdeksan kokonaislukua, joista muodostetaan käyttäjää varten etulevyyn selväkielinen virheteksti. Jokainen kokonaisluku vastaa tiettyä merkkijonoa, joka valitsimien avulla kootaan yhdeksi moniriviseksi merkkijonoksi, mikäli kyseinen kokonaisluku on nollasta poikkeava. Näin testin läpäisseet tapaukset palauttavat tyhjän merkkijonon, kun taas virhekoodin sisältävät signaalit muutetaan tekstiksi [kuva 24].



Kuva 22. Virhetekstin muodostus

Näin muodostetut virheilmoitukset tulostetaan etulevyn tuotekohtaisiin ilmoituskenttiin käyttäjää varten.

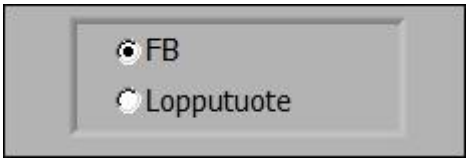
4.4 Toiminnallisuus

Alkutoimenpiteet

Testilaitteiston valmistelu aloitetaan kytkemällä käyttöjännite tietokoneeseen sekä oheislaitteisiin. Tämän jälkeen tietokoneelta käynnistetään testiohjelma, jonka jälkeen testilaitteisto on käyttövalmis.

Tuotteen tyypin valinta

Delcon Oy:n puolijohdereleet testataan kahdesti: käsiladonnan jälkeen piirilevyvaiheessa eli FB-vaiheessa sekä koteloituna lopputuotevaiheessa. FB-vaiheessa valmiit piirilevyt testataan neljän tuotteen aihioissa. Ennen tuotteen valintaa käyttäjän tulee valita, onko testattavana neljän piirilevyn aihio vai yksittäinen lopputuote. [kuva 26].

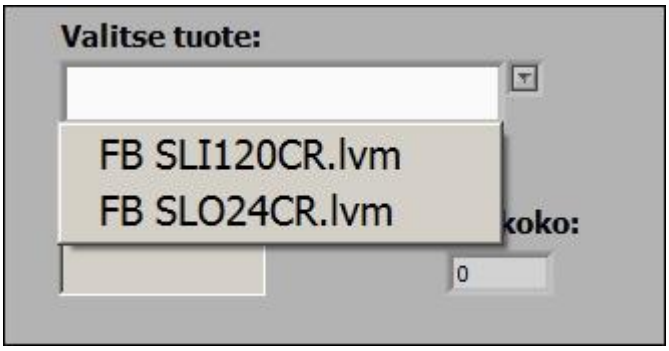


Kuva 23. Tuotteen tyypin valinta

Oletuksena lopputuotteen testauksessa on käytössä ainoastaan paikka B. neljästä testauspaikasta, mutta *admin*-välilehdeltä löytyy ylläpitoa varten nelikanavatestauksen pakottamisen mahdollistava painike. Käyttäjän valinnan mukaan sovellus suodattaa määrittelyyn sopivat raja-arvotiedostot tuotteen valinnan pudotusvalikkoon.

Tuotteen valinta

Kun tuotteen tyyppi on valittu, pudotusvalikkoon luetaan tyyppiä vastaavat raja-arvotaulukot. Näistä käyttäjän tulee valita hiirellä napsauttamalla testattavaa tuotetta vastaava tiedosto [kuva 27].



Kuva 24. Tuotteen valinta

Mikäli käyttäjä vaihtaa testattavan tuotteen tyyppiä tuotteen valinnan jälkeen, valittu tuote nollataan ja käyttäjä pakotetaan valitsemaan tuote uudestaan virhetilanteiden välttämiseksi. Raja-arvot noudetaan sarkaimin erotetusta taulukosta, joka on muodoltaan kuvan 28 kaltainen.

	A	B	C	D	E	F	G	H	I	J	K
1	Testin numero:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	Ohjausjännite	0	5	14,5	17,5	24	32	14,5	10	0	0
3	Tulovirta (yläraja)	1	5	12	12	12	13	10	9	1	0,12
4	(alaraja)	0	3	9	7,5	8	9	7	6	0	0
5	Lähtöjännite (yläraja)	65	65	65	1	1	1	1	65	65	65
6	(alaraja)	55	55	55	0	0	0	0	55	55	55
7	Lähtövirta (yläraja)	5	5	5	5	5	5	5	5	5	5
8	(alaraja)	0	0	0	0	0	0	0	0	0	3
9	LED:n tila	0	0	0	0	0	0	0	0	0	0
10	Lähtö / tulo	0									
11	Ohjaus DC / AC	0									
12	Kuorma DC / AC	0									
13	Kuorma 10 ... 16	12									
14	Kap. Häiriön mittaus	1									
15	Tuotekoht. Viive /ms	0									
16											

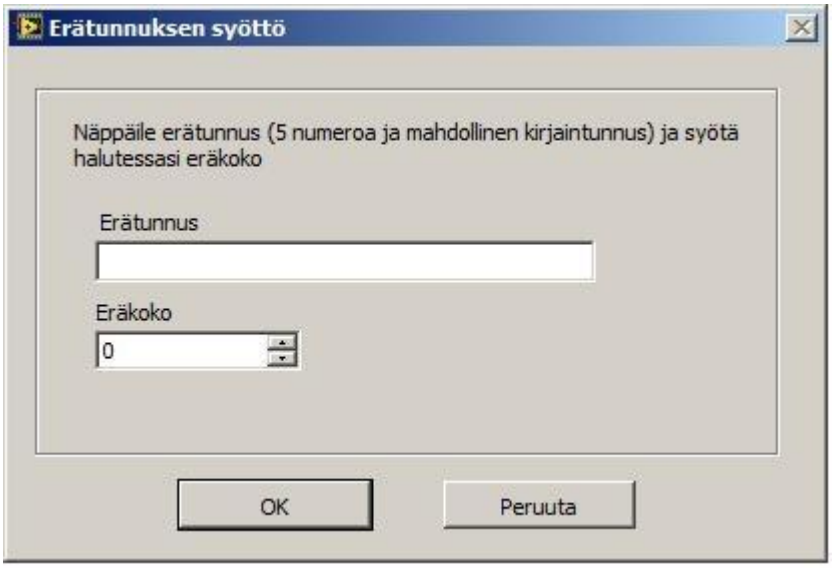
Kuva 25. Raja-arvotaulukko

Ensimmäisen testin kohdalta luetaan lisämäärityksinä releen tyyppi (lähtö/tulo), releen ohjausjännitteen tyyppi (DC/AC), kuorman tyyppi (DC/AC), kuormareleiden valinta (10–16), kapasitiivisen häiriön mittauksen valinta (0/1) ja tuotekohtainen viive

millisekunneissa. Muilta osin taulukkoa luetaan sarakkeittain aina ennen testiä, jonka jälkeen itse testirutiinissa tarkastellaan, ovatko mitatut arvot rajojen puitteissa.

Erätunnuksen ja -koon syöttö

Seuraavaksi käyttäjältä kysytään ponnahdusikkunassa testattavan tuotteen erätunnus ja valinnaisena tietona kyseisen tuote-erän eräkoko [kuva 29] Erätunnus on oltava viidestä kuuteen merkkiä pitkä, muuten ohjelma tulostaa virheilmoituksen ja pyytää syöttämään tunnuksen uudestaan.



Kuva 26. Erätunnuksen luku

Kannen sulkeminen ja testaus

Adapterin kanteen on liitetty kytkin, joka varmistaa, että adapterilaatikon kansi on kiinni eikä tilaan pääse vuotamaan valoa. Tämä on edellytys puolijohdereleen LED:n tilan luotettavalle mittaukselle. Mikäli ennakkotiedot on oikein syötetty, testausrutiini käynnistyy, kun adapterin kansi suljetaan.

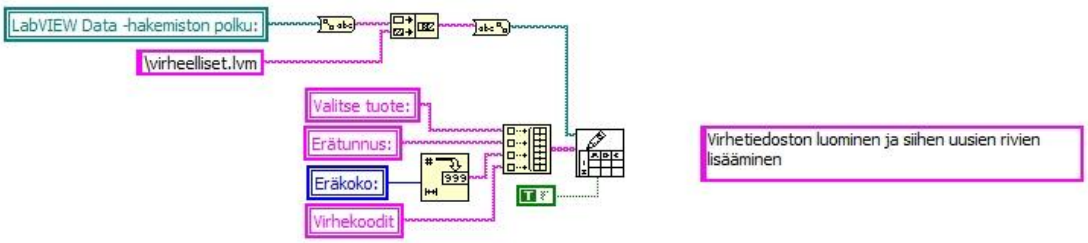
Paneelin alareunassa on neljä laatikkoa, joihin vastaavien tuotteiden ilmoitukset ilmestyvät virheen ilmetessä. Laatikot on asetettu vilkkumaan, kun virhe on havaittu kyseisen tuotteen kohdalla. Yksityiskohtainen tieto testauksen eri vaiheista, testirajoista ja ohjelmaan liittyvistä asetuksista on piilotettu *admin*-välilehdelle, jolloin perusnäkymä

on saatu pidettyä yksinkertaisena ja selkeänä. Peruskäyttäjälle oleellista on huomata virheelliset tuotteet ja poimia talteen ohjelman tulostama virhekoodi.

4.5 Tallennus

4.5.1 Mittaustulokset

Ohjelma tallentaa mittaustulokset sarkaimin erotettuun taulukkomuotoon, jonka tiedostonimi koostuu tuotteen nimestä sekä sarjanumerosta [kuva 30].



Kuva 30. Virhetaulukon luonti sekä päivitys

Taulukon luomiseen käytetty aliohjelma vaatii parametreikseen tiedostonimen päätteineen sekä absoluuttisen hakemistopolun. Funktiolle annetaan lisäksi arvoltaan ”tosi” oleva totuusarvomuuttuja, jolloin uudet tulokset lisätään jo olemassa olevan taulukon perään.

4.5.2 Virheloki

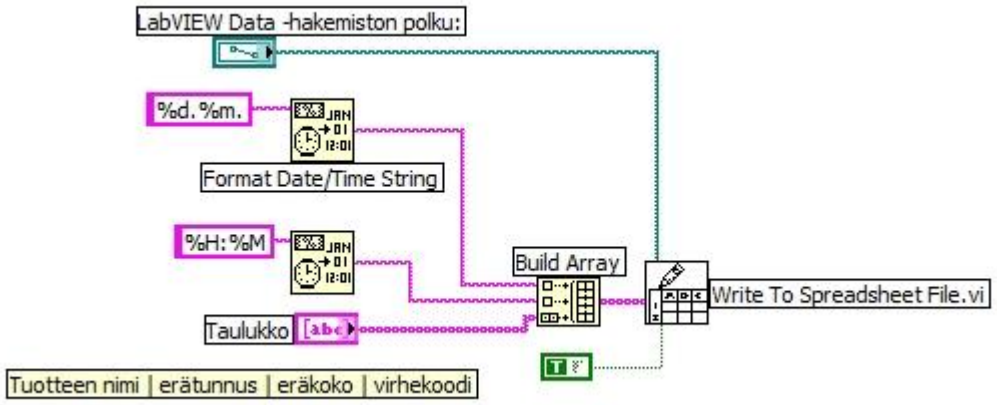
Viallisista tuotteista ylläpidetään erillistä virhelokia, johon tallennetaan päivämäärä, kellonaika, testattavana olleen tuotteen nimi, erätunnus, eräkoko sekä ohjelman luoma virhetunnus [kuva 31].

	A	B	C	D	E	F
1	pvm	klo	tuote	erätunnus	eräkoko	virhetunnus
2	11.11.	11:14	SLO24CR	12345	0	33
3	11.11.	11:25	SLO24CR	12345	0	33
4	11.11.	11:46	SLO24CR	55555	0	33
5	11.11.	15:02	SLO24CR.lvm	12345	0	33
6	12.11.	9:35	SLO24CR.lvm	50505	100	33
7	12.11.	9:39	SLO24CR.lvm	13337	80	33
8	12.11.	9:44	SLO24CR.lvm	13337	80	33

Kuva 31. Virheloki

Taulukon loppuun muodostetaan erillinen virhekoodi, joka koostuu testin numerosta (esimerkissä testi nro 3), jossa ensimmäinen virhe havaittiin, sekä numerosarjasta, joka ilmaisee virheen syyn: tulovirta liian suuri/pieni (1/2), lähtöjännite liian suuri/pieni (3/4), lähtövirta liian suuri/pieni (5/6) sekä LED:n tila väärä (7). Eli virhekoodiksi voi muodostua esimerkiksi tunnusluku 3137.

Molemmat tiedostot voidaan avata *admin*-välilehdeltä nappulan painalluksella taulukkolaskentaohjelmaan. *System Exec* -funktio käyttäytyy kuten käyttöjärjestelmissä esiintyvä komentorivikehote. Syntaksi muodostuu käytettävän taulukkolaskenta -ohjelman absoluuttisesta hakemistopolusta, jonka perään syötetään dataa sisältävän tiedoston absoluuttinen polku lainausmerkeissä [kuva 32].



Kuva 32. Virhelokin koostaminen

5 Käyttöönottokokemuksia

Suurin työmäärä testilaitteiston käyttöönotossa liittyi kanavaluettelon luontiin. Ohjelmiston testauslogiikka ja toimivuus taas voitiin testata jo aikaisemmassa vaiheessa käyttäen simuloituja mittausarvoja. Ohjelmiston muuttaminen yksikanavaisesta nelikanavaiseksi aiheutti haasteita odotettua voimakkaamman ylikuulumisen muodossa.

Ylikuuluminen on yleinen ongelma tiedonkeruukorteissa, joissa joudutaan lukemaan useita mittausarvoja monesta rinnakkaisesta kanavasta nopeaan tahtiin limittämällä. Tällöin limittäjän (eng. multiplexer) tuloportissa ollut edellinen arvo kopioituu kapasitanssin vuoksi seuraavaan mittaustulokseen. Tämä häiritsi erityisesti mitattaessa millivolttiluokan jännitetasoja. Ongelmaa yritettiin ratkaista mittausjärjestystä muuttamalla sekä lukemalla maapotentiaalia varsinaisten mittausten välissä, mutta kumpikaan ratkaisu ei tuonut toivottua lopputulosta. Mittauslaitteistosta saatiin kuitenkin viritettyä riittävän tarkka tuotannolliseen testaukseen. Haasteita aiheuttivat

muun muassa ylikuulumisesta johtuvat mittausvirheet sekä tasajännitelähteen lähdön äkillinen notkahtaminen ohjausjännitettä säädettäessä, jolloin testattavana oleva rele kytkeytyi tahattomasti pois päältä. Ongelma ratkaistiin lisäämällä kapasitanssia jännitelähteen ohjaussignaalin lähtöön, mikä hidasti kytkentäpiiriä tarpeeksi, että notkahtamisilmiö poistui.

Lopulta testilaitteisto saatiin säädettyä merkittävästi nopeammaksi verrattuna edelliseen laitteistoon. Uusi laitteisto on myös huomattavasti paremmin sekä muokattavissa että laajennettavissa tarpeen tullen. Uusien tuotteiden lisäys testilaitteistoon on tehty erittäin helpoksi, koska sovellus tutkii raja-arvohakemiston käynnistettäessä. Näin ollen uusien tuotteiden lisääminen ei vaadi kuin raja-arvotaulukon kopioinnin testipäätteelle ja sovelluksen uudelleen käynnistämisen.

6 Kehitysmahdollisuuksia

Tulevaisuudessa tuotteen valinta ja erätunnuksen syöttö olisi mahdollista hoitaa työmääräimen viivakoodia lukemalla, jolloin virheellisen tuotteen valinnan ongelmasta päästäisiin eroon. Koko testausrutiini olisi mahdollista toteuttaa myös täysautomaattisesti liukuhihnaperiaatteella robotiikkaa hyödyntäen, tällöin päästäisiin eroon suurimmasta osasta prosessia hidastavista tekijöistä, kuten adapterin kannen avaamisesta ja sulkemisesta.

Sovellusta kehitettäessä otettiin huomioon käyttäjien kokemukset edellisen laitteiston jokapäiväisestä käytöstä. Näin sovelluksen käyttöliittymä saatiin luotua peruskäyttäjän ehdoilla pelkistetyksi ja selkeäksi. Projektin loppuunsaattaminen osoittautui kokonaisuudessaan ennakoitua työläämmäksi. Myös komponenttien asettelu koteloihin, johdottaminen sekä kiinnitys vaati rutkasti aikaa ja huolellisuutta. LabVIEW-sovelluksen yhdistäminen mittaus-kytkentöihin oli tämän jälkeen suoraviivainen operaatio.

Työn vaatimuksena oli kehittää helppokäyttöinen testausohjelmisto Delcon-puolijohdevälireleiden testaukseen elektroniikkatuotannon työntekijöille. Testilaitteiston on määrä palvella käyttäjiä nopeuttaen jokapäiväistä testausurakkaa. Projekti oli osa Delcon Oy:n tuotannon tehostamista, jolla Delcon Oy pystyy säilyttämään asemansa markkinajohtajana omalla segmentillään sekä vastaamaan tämänhetkiseen tukalaan taloustilanteeseen entistä tehokkaammalla tuotantoprosessilla.

Lähteet

1 LabVIEW Graphical Development Hands-On Seminar, Customer Manual, October 2005 Edition – Update May 2006, Northern Region Version, Luettu 1.9.2007.

2 LabVIEW Fundamentals, Version 8.0, National Instruments, August 2005, Luettu 2.6.2009.

3 Data Acquisition (DAQ) Hardware. (WWW-dokumentti.) National Instruments. <http://www.ni.com/daq/> Luettu 2.6.2009.

4 Delcon Oy: Electrical Design (WWW-dokumentti.) <http://www.delcon.fi/delcon/Technology/Electricaldesign/tabid/72/language/fi-FI/Default.aspx> Luettu 4.6.2009.

5 Mäki-Tanila, Jouni. Tuotantokäsikirja (TOK), Versio 3, Delcon Oy, sisäinen dokumentti, 8. tammikuuta 2007, Luettu 1.10.2008.

6 Temmes Ilkka, Artikkeli sähköistysjärjestelmistä, Automaatioväylä-lehti, 6/2008, Luettu 1.10.2008.

7 Konttinen Reima, Tuotekäsikirja (TEK1), Versio 2.4, Delcon Oy, sisäinen dokumentti, 17. heinäkuuta 2007, Luettu 1.10.2008.

8 Konttinen Reima, Laatukäsikirja, Versio 1.0, Delcon Oy, sisäinen dokumentti, 23. elokuuta 2002, Luettu 1.10.2009